Lab 10: Predator-Prey Simulation

due: November 30nd or Dec 1st, 2017

Predator-prey simulations are used to understand how populations of animals interact. In this lab's agent-based simulation, there will be two types of animals: Prey and Predator, for example, rabbits and wolves. Your task is to program the individual animal behaviors. Both types of animals run around randomly and reproduce with some probability. The predators eat nearby prey, removing prey from the population. The predators gain energy from eating the prey, but will die if their energy drops below zero. Animals are born with some amount of energy, and predators expend energy hunting each timestep. The predators are also less likely to reproduce if their energy is low.

The `Animal` parent class and the `World` class are provided, in addition to the main sketch. Your task is to implement the `Prey` and `Predator` child classes. You should not modify the `World` or `Animal` classes. Resist the urge to copy and paste**; use inheritance**. Next, run experiments with your simulation. Gather data about how the populations change as you vary your parameters: initial population size, speed, movement strategy, reproduction rate. Try exchanging your classes with a classmate.

Learning Objectives

- ❑ Use inheritance
- ❑ Explore agent-based simulation.
- ❑ Write more Java classes.

Deliverables

- ❑ Your program should start with a comment that includes your name, email, date, assignment description, collaboration statement, and reflection.
- ❑ Bring a hardcopy of your program (the source code, not the graphics) to your next lab period.
- ❑ Be prepared to run the Processing sketch and demonstrate your "Theory of the Program."

```
World w;

void setup() {
  size(500, 500);
  w = new World();
  for (int i = 0; i < 50; i++) {
     w.addAnimal(new Prey(w));
  }
  for (int i = 0; i < 10; i++) {
     w.addAnimal(new Predator(w));
  }
}

void draw() {
  background(196);
  w.tick();
  drawPop();
}

void drawPop() {
  int numPrey = w.count(Prey.class);
  int numPredator = w.count(Predator.class);
  if (numPrey == 0 || numPredator == 0) {
    background(196);
    textSize(32); fill(24);
    text("steps: " + frameCount, 50, 50);
    text("prey: " + numPrey, 50, 100);
    text("predator: " + numPredator, 50, 150);
    noLoop();
  } else {
    noStroke(); fill(0, 0, 255, 124);
    rect(0, 10, numPrey*3, 18);
    fill(255, 0, 0, 124);
    rect (0, 30, numPredator*3, 18);
  }
}
```

```
class Animal {
  float x, y, sz;
  float fspeed, tspeed;
  float heading;
  float energy;
  boolean alive = true;
  float reproductionRate = 0.01;
  World w;

  Animal(World w) {
    x = random(width);
    y = random(height);
    tspeed = 0.1;
    fpseed = 2;
    sz = 15;
    energy = 200;
    heading = random(2*PI);
    this.w = w;
  }

  void display() {}

  void show() {
    if (alive) {
      pushMatrix();
      translate(x, y);
      rotate(heading);
      display();
      popMatrix();
    }
  }

  boolean shouldReproduce() {
    return random(1) < reproductionRate;
  }
```

```
void turn(float dRads) {
  heading += dRads;
}

void forward(float amount) {
  x += amount * cos(heading);
  y += amount * sin(heading);
}

void move() {
  turn(random(-tspeed, tspeed));
  forward(random(fspeed));
}

void step() {
  move();
  fence();
}

void wrap() {
  if (x > width) x = 0;
  else if (x < 0 ) x = width;

  if (y > height) y = 0;
  else if (y < 0 ) y = height;
}

void fence() {
  x = constrain(x, 0, width);
  y = constrain(y, 0, height);
}

void die() {
  alive = false;
}
```

```
  float distance(Animal a) {
    return dist(x, y, a.x, a.y);
  }
}
```

```
class World {
  private final int MAX_ANIMALS = 10000;
  private Animal[] animals;
  private int num = 0;

  World() {
    animals = new Animal[MAX_ANIMALS];
  }

  void tick() {
    for (int i = 0; i < num; i++) {
      animals[i].step();
      animals[i].show();
    }
    removeDead();
  }

  void addAnimal(Animal a) {
    animals[num] = a;
    num++;
  }

  void removeDead() {
    for (int i = num-1; i >= 0; i--) {
      if (!animals[i].alive) {
        animals[i] = animals[num-1];
        animals[num-1] = null;
        num--;
      }
    }
  }
```

```
  int count(Class T) {
    int count = 0;
    for (int i = 0; i < animals.length; i++) {
      if (T.isInstance(animals[i])) {
        count ++;
      }
    }
    return count;
  }

  boolean nearBy(Animal c, float maxD, Class T) {
    for (int i = 0; i < num; i++) {
      float d =  c.distance(animals[i]);
      if (d < maxD && T.isInstance(animals[i])) {
        return true;
      }
    }
    return false;
  }

  Animal nearBy(Animal c) {
    if (num <= 0) return null;
    Animal closest = animals[0];
    float minDist =  c.distance(animals[0]);
    for (int i = 1; i < num; i++) {
      float d = c.distance(animals[i]);
      if (c != animals[i] && d < minDist) {
        minDist = d;
        closest = animals[i];
      }
    }
    return closest;
  }
}
```